

# XSD-XML

## Что такое XML?

- **XML** - аббревиатура от англ. *eXtensible Markup Language* (пер. расширяемый язык разметки).
- **XML** – язык разметки, который напоминает HTML.
- **XML** предназначен для передачи данных, а не для их отображения.
- **Теги XML** не predetermined. Вы должны сами определять нужные теги.
- **XML** описан таким образом, чтобы быть самоопределяемым.

## Что такое XML элемент

**XML элемент** — это все от (и включая) начального тега элемента до (и включая) конечного тега элемента.

Элемент может содержать:

- другие элементы
- текст
- атрибуты
- или набор из всего выше названного

```

<bookstore>
  <book category="CHILDREN">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>

```

В приведенном выше примере **<bookstore>** и **<book>** содержат элементный контент, состоящий из других элементов. Также, у **<book>** есть атрибут (*category="CHILDREN"*). У элементов **<title>**, **<author>**, **<year>** и **<price>** есть текстовый контент.

## Пустые XML элементы

При написании элементов без контента можно использовать альтернативный синтаксис.

Вместо того, чтобы писать пустой элемент в виде: `<book></book>`, можно написать: `<book />`

## Правила написания имен XML

XML элементы должны следовать следующим правилам написания имен:

- Имена могут содержать буквы, числа и другие символы
- Имена не могут начинаться с цифры или символа пунктуации
- Имена не могут начинаться с сочетания "xml" (или XML, или Xml и т.п.)
- Имена не могут содержать пробельные символы

\*В качестве имен можно использовать любые слова. Нет зарезервированных слов.

# Хорошая практика составления имен

Старайтесь придумать описательные имена: `<first_name>`, `<last_name>`.

Имена следует составлять короткие и простые, вроде: `<book_title>`; а не: `<the_title_of_the_book>`.

Избегайте символ "-". Если вы напишете нечто вроде "first-name", то некоторые приложения могут решить, что вы вычитаете имя "name" из имени "first".

Избегайте символ ".". Если вы напишете нечто вроде "first.name", то некоторые приложения могут решить, что "name" это свойство объекта "first".

Избегайте символ ":". Двоеточие зарезервировано для механизма пространства имен.

Не-латинские символы, вроде, ёдѧ вполне легальны в XML, однако, если некое приложение их не поддерживает, то возникнут проблемы.

## Стили написания имен

Для XML элементов не существует какого-либо определенного стиля написания имен. Тем не менее, вот несколько наиболее часто используемых правил использования стилей:

Если вы выбрали какой-либо стиль написания имен, то следует последовательно придерживаться его!

Очень часто XML документ сопровождается соответствующей базой данных. Хорошей практикой является использование таких же правил написания имен элементов XML документа, что и в соответствующей базе данных.

## Расширяемость XML элементов

XML элементы могут быть расширены, чтобы нести больше информации.

### Пример:

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <body>Не забудь про меня в эти выходные!</body>
</note>
```

Давайте представим, что мы создали приложение, которое извлекает элементы **<to>**, **<from>** и **<body>** из XML документа и формирует следующее сообщение:

```
СООБЩЕНИЕ
Кому: Tove
От: Jani
Не забудь про меня в эти выходные!
```

Представьте, что автор XML документа добавил некоторую дополнительную информацию:

```
<note>
  <date>2008-01-10</date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Напоминание</heading>
  <body>Не забудь про меня в эти выходные!</body>
</note>
```

Прервется ли работа нашего приложения?

Нет. Приложение все равно будет способно отыскать элементы **<to>**, **<from>** и **<body>** и сформировать тот же самый вывод.

Одно из главных достоинств XML состоит в том, что XML документ можно легко расширять не влияя на работу исходного приложения.

## XML атрибуты

Атрибуты предоставляют дополнительную информацию об элементе. Атрибуты часто предоставляют информацию, не являющуюся частью данных, но эта информация может быть важна для приложений, которые будут манипулировать этим элементом:

пример

```
<file type="gif">computer.gif</file>
```

## XML атрибуты должны заключаться в кавычки

Значение атрибута всегда должно заключаться в кавычки. Это могут быть либо двойные, либо одинарные кавычки. Например

```
<person sex="female"> или <person sex='female'>
```

Если значение атрибута само содержит двойные кавычки, то можно использовать одинарные кавычки. Например:

```
<gangster name='George "Shotgun" Ziegler'>
```

либо использовать символы сущностей:

```
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

При использовании атрибутов возникают некоторые проблемы:

- атрибуты не могут содержать множественные значения (элементы могут)
- атрибуты не могут содержать древовидные структуры (элементы могут)
- атрибуты сложно расширять (для будущих изменений)

Атрибуты нелегко читать и обслуживать. Используйте элементы для данных, а атрибуты для информации, не относящейся к данным.

## Теги

В XML каждый элемент должен быть заключен в теги. Тег — это некий текст, обернутый в угловые скобки: `<tag>`

Текст внутри угловых скобок — название тега.

Тега всегда два:

- Открывающий — текст внутри угловых скобок: `<tag>`
- Закрывающий — тот же текст (это важно!), но добавляется символ «/»:

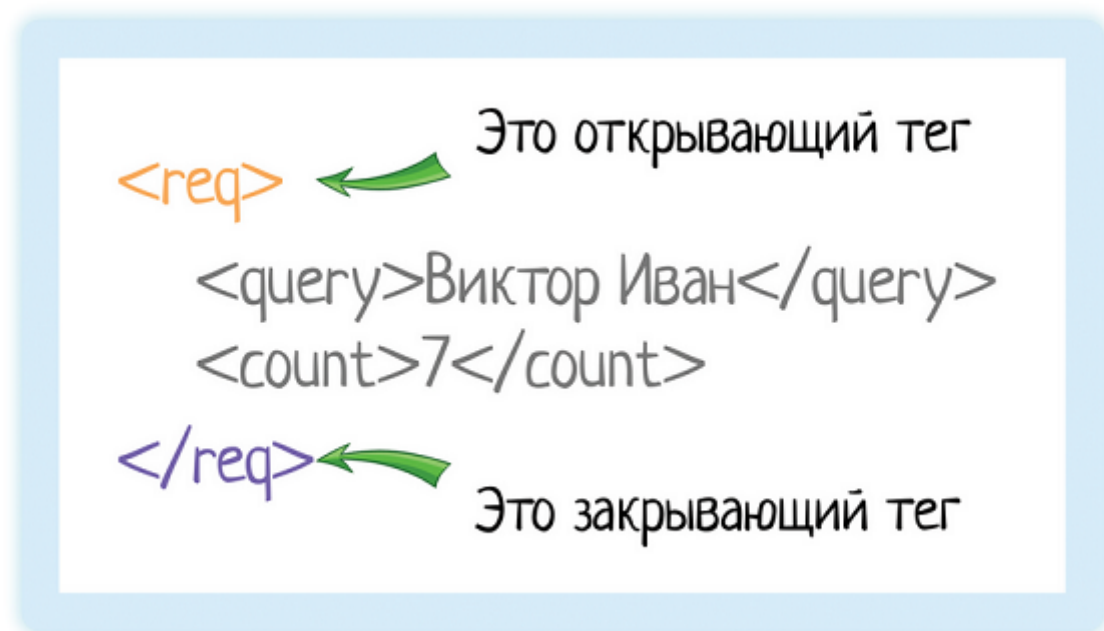
```
</tag>
```

Пустые XML элементы

При написании элементов без контента можно использовать альтернативный синтаксис.

Вместо того, чтобы писать пустой элемент в виде: `<book></book>`, можно написать: `<book />`

### Пример:

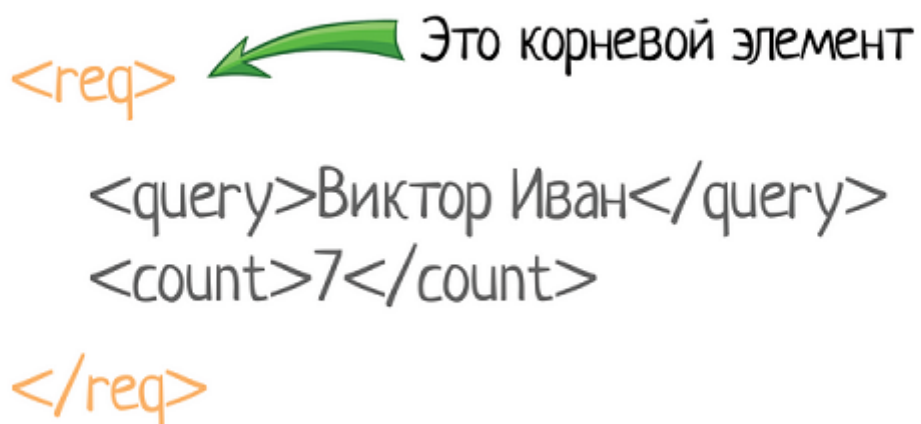


С помощью тегов мы показываем системе «вот тут начинается элемент, а вот тут заканчивается».

### **Корневой элемент**

В любом XML-документе есть корневой элемент. Это тег, с которого документ начинается, и которым заканчивается. В случае REST API документ — это запрос, который отправляет система. Или ответ, который она получает.

Чтобы обозначить этот запрос, нам нужен корневой элемент.



`<req>` ← Это корневой элемент

```
<query>Виктор Иван</query>  
<count>7</count>
```

`</req>`

Он показывает начало и конец нашего запроса, не более того. А вот внутри уже идет тело документа — сам запрос. Те параметры, которые мы передаем внешней системе. Разумеется, они тоже будут в тегах, но уже в обычных, а не корневых.

### Значение элемента

Значение элемента хранится между открывающим и закрывающим тегами. Это может быть число, строка, или даже вложенные теги!

Вот у нас есть тег «query». Он обозначает запрос, который мы отправляем в подсказки.

Тег скажет системе, что внутри  
хранится запрос для подсказок

```
<req>  
  <query>Виктор Иван</query>  
  <count>7</count>  
</req>
```

Внутри — значение запроса.

Значение запроса

```
<req>  
  <query>Виктор Иван</query>  
  <count>7</count>  
</req>
```

Системе надо как-то передать, что «пользователь ввел именно это». Как показать ей, где начинается и заканчивается переданное значение? Для этого и используются теги.

Система видит тег «query» и понимает, что внутри него «строка, по которой нужно вернуть подсказки».



## Теги

